Apple Assembly Line

$1.50

Volume 2 -- Issue 7 April, 1982

In This Issue...

Adding Auto-SAVE to S-C Macro Assembler 2
Review of AED][(A new Applesoft Editor) 10
Ashby's Easy Shift-Key Modifier 13
Potential Trouble in TYMAC 15
Using Macros and Nested Macros 17
Recursive Macros 22
Controlling Software Configuration 24
Funny Noise 27

Another New Book: Bag of Tricks

The authors of Beneath Apple DOS (Don Worth and Pieter Lechner) have done it again! This time you get a diskette with four powerful disk utilities on it, and a book expaining their use. The retail price is $39.95, but I will have them for only $36.

The utilities are TRAX, INIT, ZAP, and FIXCAT. TRAX examines any track on a disk, reading it in as pure nibbles and displaying in a partially analyzed form. INIT reformats any track or tracks, optionally retaining existing data in whatever readable sectors are in the track. You can reorder the sectors, change the volume number, and more. ZAP is a general purpose disk utility: sectors may be read, written, displayed, modified with a powerful assortment of over 50 commands. It works with 13- and 16-sector DOS, as well as Pascal and CP/M diskettes. You can even "program" in ZAP, with labels, loops, and macro-commands. FIXCAT can automatically repair or reconstruct a catalog track by analyzing the rest of the disk.

Beyond the utilities themselves, there is about 40 pages of advanced tutorial material which starts where "Beneath Apple DOS" ends.

Unless you are fully satisfied with your present collection of disk utilities, you ought to get this set.

Adding Auto-SAVE to S-C Macro Assembler.....Greg H. Anders

[Greg is a subscriber from Albuquerque, New Mexico.]

One of the nice features of the new S-C Macro Assembler is the title directive (.TI). This directive causes a title and page number to be printed at the top of each page of an assembly listing. The title directive gave me the idea for the Automatic Save command program which follows.

I felt the need for an Auto Save command because of my own carelessness. After extensive editing of a rather lengthy program, I decided it was a good time to save the program before I proceeded. The file names I use are usually descriptive and forgettable, so to save a file, I list the Catalog, then use the cursor controls to copy the file name. After the file name appeared on the screen, I zipped the cursor next to the name I wanted to save the file under and, succumbing to temporary insanity, typed an "L". The word "LOAD" flashed on the screen and my mouth dropped open in disbelief. The only sounds that could be heard were the whirr of the disk drive and the screams of my new code byting the dust cover!

I decided to try to simplify the task of saving a program, giving myself less chance of making an error. From this came the Auto Save command. With this command, typing SAVE does not save your program on cassette. Instead, the SAVE command searches your source program for a title. If a title is found and it is a valid DOS name, the source program is automatically saved, using the title as the file name. In addition, if you end your title with a version number in the form N.N, Auto Save automatically increments the version number in the source program and saves the program using the new version number. The version number option does not erase your old file, which means your old file is a back-up. Be careful, though. A few saves and your disk is full of back-up files. You'll need to go back and delete a file or two every once in a while.

The version number goes up to a maximum of 9.9, after which it starts back at 0.0. If the version number option is not desired, don't put a number in the form N.N at the end of your title.

Leading and trailing blanks are ignored by Auto Save. If there is more than one consecutive blank in a title, the blanks are compressed to one. Thus, the title ".TI 56,TI TLE" generates a SAVE to the file named "TI TLE". Also, any commas in your title are changed to dashes so as not to confuse DOS.

To use the Auto Save command, the vector address of the SAVE command must be changed. The address must be one less than the actual start of the Auto Save command. For example, if Auto Save is assembled at $800, the address would be changed in the table inside the S-C Macro Assembler to $07FF.

For the version of the S-C Macro Assembler which loads at $1000, change the contents of address $1679 to $07 and $1678 to $FF. Shown as a monitor command, this would be:

```
:$1678:FF 07
```

For the Language Card version of the S-C Macro Assembler, change the content of address $D679 to $07 and $D678 to $FF. You have to write-enable the card first:

```
:$C083 C083 D678:FF 07
```

I like to keep Auto Save behind the Language Card version of the Macro Assembler. I put the program at $F320 and the changes are:

```
:$C083 C083 D678:1F F3
```

One thing you'll have to look out for. If you type an illegal DOS SAVE command such as "SAVE 1 4 THE ROAD", DOS ignores this command and the Auto Save goes into effect; the "1 4 THE ROAD" is ignored. Also note that the save is performed on the drive that is active. Since commas are changed to dashes, there is currently no way to specify which drive you want the save to be performed on. Perhaps you would like to try to implement this enhancement yourself.

After you've installed the Auto Save program, type in this program:

```
1000 *    A TEST OF AUTO SAVE
1010      .TI 54, TITLE TEST VER. 0.9
```

Then type SAVE, and CATALOG. See how the file was saved? List the file and notice the change in line 1010. Voila!

For those of you who haven't updated to the Macro Assembler yet, Auto Save can be implemented with S-C Version 4.0 by using the .US command for the title. The changes which are necessary are outlined below.

1. The following lines must be deleted: 1490-1540, 2090-2150, 2460-2470, 2560-2930.

2. The following lines must be added:

```
1210        .US S-C VER. 4.0 AUTO SAVE 1.0
1600        BNE .2        ...ALWAYS
1920 *   CHECK THE OP CODE FOR .US
2170        BCS TITLE
2480 .1     CMP #$80
3480 OPS    .AS /.US/
3510 NO.TTL .AS /*** NO TITLE ERRO/
3515        .AS  -/R/
3520        .AS /*** ILLEGAL TITLE FIRST CHARACTE/
3525        .AS  -/R/
```

3. Change the SAVE vector address. For an origin of $800, that would be

```
:$1271:FF 07
```

4. To use the command, put the title you want to use for the file name like so:

```
.US MY TITLE VER. 1.0
```

```
                    1000 *--------------------------------
                    1010 *      AUTOMATIC SAVE PROGRAM
                    1020 *      THIS PROGRAM CHECK'S FOR A TITLE
                    1030 *      AND IF ONE IS FOUND, THE CURRENT PROGRAM
                    1040 *      IS SAVED UNDER THE TITLE
                    1050 *      ALSO, IF THE VERSION NUMBER IS APPENDED
                    1060 *      IT IS UPDATED BEFORE EACH SAVE
                    1070 *--------------------------------
                    1080 *      SYSTEM EQUATES
                    1090 *--------------------------------
FDED-               1100 MON.COUT   .EQ $FDED
FD8E-               1110 MON.CROUT  .EQ $FD8E
FBDD-               1120 MON.BELL1  .EQ $FBDD
0200-               1130 IN.BUF     .EQ $200
004C-               1140 SRC.END    .EQ $4C,4D
00CA-               1150 SRC.START  .EQ $CA,CB
001D-               1160 NEXT       .EQ $1D
001E-               1170 SEARCH     .EQ $1E,1F
                    1180 *--------------------------------
                    1190         .OR $800
                    1200         .TF AUTO.SAVE.OBJECT A$800
                    1210 *--------------------------------
                    1220 *      INITIALIZE SEARCH REGISTERS AND
                    1230 *      DETERMINE IF AT END OF SOURCE PROGRAM
                    1240 *--------------------------------
                    1250 AUTO.SAVE
0800- A5 CA         1260         LDA SRC.START      GET START OF SOURCE PROGRAM ADDRESS
0802- 85 1E         1270         STA SEARCH         AND MOVE TO THE SEARCH ADDRESS
0804- A5 CB         1280         LDA SRC.START+1                    REGISTER
0806- 85 1F         1290         STA SEARCH+1
0808- D8            1300         CLD
                    1310 ADDRESS.END.CMP
0809- A5 1E         1320         LDA SEARCH
080B- C5 4C         1330         CMP SRC.END        SEE IF AT END OF SOURCE PROGRAM
080D- D0 06         1340         BNE .1
080F- A5 1F         1350         LDA SEARCH+1
0811- C5 4D         1360         CMP SRC.END+1
0813- F0 2F         1370         BEQ ERROR1         DIDN'T FIND TITLE
                    1380 *--------------------------------
                    1390 *      SEARCH LINE FOR OP CODE
                    1400 *--------------------------------
0815- A0 00         1410 .1      LDY #0         Y OFFSET FOR LINE EXAMINATION
0817- B1 1E         1420         LDA (SEARCH),Y     NEXT LINE OFFSET
0819- 85 1D         1430         STA NEXT
081B- A0 03         1440         LDY #3         POINT TO CHARACTER AFTER LINE NUMBER
081D- B1 1E         1450         LDA (SEARCH),Y
081F- C9 2A         1460         CMP #'*        COMMENT LINE?
0821- F0 14         1470         BEQ NEW.LINE       YEP
0823- C9 C0         1480 .5      CMP #$C0       COMPRESSED CODE?
0825- D0 05         1490         BNE .2         NOPE
0827- C8            1500 .4      INY            MOVE OFFSET PAST COMPRESSED INFO
0828- C8            1510         INY
0829- B8            1520         CLV
082A- 50 04         1530         BVC .3         ...ALWAYS
082C- C9 80         1540 .2      CMP #$80       SPACE(S)?
082E- B0 34         1550         BCS OPCHK      YES, CHECK THE OP-CODE
0830- C8            1560 .3      INY
0831- B1 1E         1570         LDA (SEARCH),Y
0833- F0 02         1580         BEQ NEW.LINE       END OF LINE (EOL) IS 0
0835- D0 EC         1590         BNE .5         ...ALWAYS
                    1600 *--------------------------------
                    1610 *      CALCULATE ADDRESS OF NEXT LINE
                    1620 *--------------------------------
                    1630 NEW.LINE
0837- 18            1640         CLC
0838- A5 1E         1650         LDA SEARCH     MOVE SEARCH ADDRESS TO NEXT LINE
083A- 65 1D         1660         ADC NEXT
083C- 85 1E         1670         STA SEARCH
083E- 90 C9         1680         BCC ADDRESS.END.CMP
0840- E6 1F         1690         INC SEARCH+1
0842- D0 C5         1700         BNE ADDRESS.END.CMP    ...ALWAYS
```

```
                     1710 *--------------------------------
                     1720 *       ERROR ROUTINES
                     1730 *--------------------------------
                     1740 ERROR1
0844- A0 00          1750         LDY #0        POINT TO NO TITLE ERROR
0846- B9 54 09       1760 PRTERR  LDA NO.TTL,Y
0849- 30 08          1770         BMI ERREND
084B- 09 80          1780         ORA #$80
084D- 20 ED FD       1790         JSR MON.COUT
0850- C8             1800         INY
0851- D0 F3          1810         BNE PRTERR
0853- 20 ED FD       1820 ERREND  JSR MON.COUT
0856- 20 DD FB       1830         JSR MON.BELL1
0859- 20 DD FB       1840         JSR MON.BELL1
085C- 20 8E FD       1850         JSR MON.CROUT
085F- 60             1860         RTS
                     1870 ERROR2
0860- A0 12          1880         LDY #18       POINT TO ILLEGAL CHAR. ERROR
0862- D0 E2          1890         BNE PRTERR    ...ALWAYS
                     1900 *--------------------------------
                     1910 *       CHECK THE OP CODE FOR .TI
                     1920 *--------------------------------
0864- A2 00          1930 OPCHK   LDX #0
0866- C8             1940 .1      INY
0867- B1 1E          1950         LDA (SEARCH),Y
0869- F0 CC          1960         BEQ NEW.LINE       EOL
086B- DD 4C 09       1970         CMP OPS,X     COMPARE OP CODE
086E- D0 C7          1980         BNE NEW.LINE THAT'S NOT IT
0870- E8             1990         INX
0871- E0 03          2000         CPX #3        IF ALL 3 COMPARE, FOUND OP CODE
0873- D0 F1          2010         BNE .1
                     2020 *--------------------------------
                     2030 *       NOW LOOK FOR TITLE
                     2040 *--------------------------------
0875- C8             2050 TITLE   INY
0876- B1 1E          2060         LDA (SEARCH),Y
0878- F0 CA          2070         BEQ ERROR1    NO TITLE?
087A- C9 2C          2080         CMP #',       LOOKING FOR COMMA (TITLE FOLLOWS)
087C- D0 F7          2090         BNE TITLE
087E- C8             2100 .1      INY
087F- B1 1E          2110         LDA (SEARCH),Y
0881- F0 C1          2120         BEQ ERROR1    NO TITLE?
0883- C9 C0          2130         CMP #$C0      COMPRESSED?
0885- F0 40          2140         BEQ COMP.CODE1
0887- C9 80          2150         CMP #$80      SPACE?
0889- B0 F3          2160         BCS .1        YEP--SKIP
088B- C9 41          2170         CMP #'A       MAKE SURE 1ST CHAR. IS LETTER
088D- 90 D1          2180         BCC ERROR2    NOT LETTER
088F- C9 5B          2190         CMP #$5B      1 MORE THAN "Z"
0891- B0 CD          2200         BCS ERROR2
                     2210 *--------------------------------
                     2220 *       TITLE FOUND
                     2230 *       OUTPUT CTRL-D, "SAVE" AND TITLE
                     2240 *--------------------------------
0893- 48             2250         PHA
0894- A2 00          2260         LDX #0
0896- BD 4F 09       2270 .2      LDA SAVE,X
0899- 20 ED FD       2280         JSR MON.COUT
089C- E8             2290         INX
089D- E0 05          2300         CPX #5
089F- D0 F5          2310         BNE .2
08A1- 68             2320         PLA
                     2330 NEXT.CHAR1
08A2- 09 80          2340         ORA #$80
08A4- 20 ED FD       2350         JSR MON.COUT
08A7- E8             2360         INX           X KEEPS TRACK OF INPUT BUFFER OFFSET
                     2370 NEXT.CHAR2
08A8- C8             2380         INY
08A9- B1 1E          2390         LDA (SEARCH),Y
08AB- F0 56          2400         BEQ GOT.TTL2  EOL--GOT THE TITLE
08AD- C9 2C          2410         CMP #',       NO COMMAS ALLOWED
08AF- D0 04          2420         BNE .1
08B1- A9 2D          2430         LDA #'-       REPLACE COMMA WITH DASH
```

```
08B3- D0 ED     2440         BNE NEXT.CHAR1     ...ALWAYS
08B5- C9 C0     2450 .1      CMP #$C0
08B7- F0 2F     2460         BEQ COMP.CODE2
08B9- C9 80     2470         CMP #$80
08BB- 90 E5     2480         BCC NEXT.CHAR1
08BD- C8        2490         INY             CHECK FOR CHARACTER AFTER SPACE
08BE- B1 1E     2500         LDA (SEARCH),Y
08C0- F0 40     2510         BEQ GOT.TTL1 DROP TRAILING SPACES
08C2- 88        2520         DEY             MOVE POINTER BACK TO CORRECT POSITION
08C3- A9 20     2530         LDA #$20        SPACE--SPACES IN TITLE COMPRESSED TO 1
08C5- D0 DB     2540         BNE NEXT.CHAR1     ...ALWAYS
                2550 *-------------------------------
                2560 *       COMPRESSED CHARACTER ROUTINES
                2570 *-------------------------------
                2580 COMP.CODE1
08C7- C8        2590         INY
08C8- B1 1E     2600         LDA (SEARCH),Y     THIS IS NUMBER OF CHARACTERS
08CA- 85 1D     2610         STA NEXT                              COMPRESSED
08CC- C8        2620         INY
08CD- B1 1E     2630         LDA (SEARCH),Y     ACTUAL CHARACTER
08CF- C9 41     2640         CMP #'A         MAKE SURE IT'S A LETTER
08D1- 90 8D     2650         BCC ERROR2
08D3- C9 5B     2660         CMP #$5B
08D5- B0 89     2670         BCS ERROR2
08D7- 48        2680         PHA
08D8- A2 00     2690         LDX #0
08DA- BD 4F 09  2700 .1      LDA SAVE,X
08DD- 20 ED FD  2710         JSR MON.COUT
08E0- E8        2720         INX
08E1- E0 05     2730         CPX #5
08E3- D0 F5     2740         BNE .1
08E5- 68        2750         PLA
08E6- D0 0E     2760         BNE STORE       ...ALWAYS
                2770 COMP.CODE2
08E8- C8        2780         INY
08E9- B1 1E     2790         LDA (SEARCH),Y
08EB- 85 1D     2800         STA NEXT
08ED- C8        2810         INY
08EE- B1 1E     2820         LDA (SEARCH),Y
08F0- C9 2C     2830         CMP #'
08F2- D0 02     2840         BNE STORE
08F4- A9 2D     2850         LDA #'-
                2860 STORE
08F6- 09 80     2870         ORA #$80
08F8- 20 ED FD  2880         JSR MON.COUT
08FB- E8        2890         INX
08FC- C6 1D     2900         DEC NEXT
08FE- D0 F6     2910         BNE STORE
0900- F0 A6     2920         BEQ NEXT.CHAR2
                2930 *-------------------------------
                2940 *       SEARCH FOR VERSION NUMBER AND CHANGE IF FOUND
                2950 *-------------------------------
                2960 GOT.TTL1
0902- 88        2970         DEY
                2980 GOT.TTL2
0903- 88        2990         DEY             MOVE Y POINTER TO THIRD NON-BLANK
0904- 88        3000         DEY             CHARACTER FROM THE END OF LINE
0905- 88        3010         DEY
0906- CA        3020         DEX
0907- B1 1E     3030         LDA (SEARCH),Y     THIRD CHAR. FROM END
0909- C9 30     3040         CMP #'0
090B- 90 3B     3050         BCC DOS.OP
090D- C9 3A     3060         CMP #':         ASCII ":" IS 1 MORE THAN ASCII 9
090F- B0 37     3070         BCS DOS.OP
0911- C8        3080         INY
0912- B1 1E     3090         LDA (SEARCH),Y     2ND CHAR. FROM END
0914- C9 2E     3100         CMP #'.         SHOULD BE PERIOD
0916- D0 30     3110         BNE DOS.OP
0918- C8        3120         INY
0919- B1 1E     3130         LDA (SEARCH),Y     LAST CHARACTER
091B- C9 30     3140         CMP #'0
091D- 90 29     3150         BCC DOS.OP
091F- C9 3A     3160         CMP #':
```

```
0921- B0 25     3170         BCS DOS.OP
0923- 69 01     3180         ADC #1
0925- C9 3A     3190         CMP #':
0927- D0 18     3200         BNE STORIT
0929- A9 30     3210         LDA #'0
092B- 91 1E     3220         STA (SEARCH),Y    CHANGE DIGIT IN SOURCE CODE
092D- 09 80     3230         ORA #$80
092F- 9D 00 02  3240         STA IN.BUF,X      CHANGE DIGIT IN DOS COMMAND
0932- CA        3250         DEX
0933- CA        3260         DEX
0934- 88        3270         DEY
0935- 88        3280         DEY
0936- B1 1E     3290         LDA (SEARCH),Y
0938- 18        3300         CLC
0939- 69 01     3310         ADC #1
093B- C9 3A     3320         CMP #':
093D- D0 02     3330         BNE STORIT
093F- A9 30     3340         LDA #'0
0941- 91 1E     3350 STORIT  STA (SEARCH),Y
0943- 09 80     3360         ORA #$80
0945- 9D 00 02  3370         STA IN.BUF,X
                3380 *-------------------------------
                3390 *       CR OUTPUT CAUSES DOS TO PERFORM SAVE
                3400 *       AFTERWARDS, RETURN TO ASSEMBLER
                3410 *-------------------------------
0948- 20 8E FD  3420 DOS.OP  JSR MON.CROUT
094B- 60        3430 END     RTS
                3440 *-------------------------------
                3450 *       MESSAGES
                3460 *-------------------------------
094C- 2E 54 49  3470 OPS     .AS /.TI/
094F- 84        3480 SAVE    .HS 84        CTRL-D
0950- D3 C1 D6
0953- C5        3490         .AS -/SAVE/
0954- 2A 2A 2A
0957- 20 4E 4F
095A- 20 54 49
095D- 54 4C 45
0960- 20 45 52
0963- 52 4F D2  3500 NO.TTL  .AT /*** NO TITLE ERROR/
0966- 2A 2A 2A
0969- 20 49 4C
096C- 4C 45 47
096F- 41 4C 20
0972- 54 49 54
0975- 4C 45 20
0978- 46 49 52
097B- 53 54 20
097E- 43 48 41
0981- 52 41 43
0984- 54 45 D2  3510         .AT /*** ILLEGAL TITLE FIRST CHARACTER/
0987-           3520 ZZZEND  .EQ *
0187-           3530 ZZZLEN  .EQ ZZZEND-AUTO.SAVE
```

AED -- A New Applesoft Program Editor.....Reviewed by
Bob Sander-Cederlof

One of the joys of putting the Apple Assembly Line out each month has been the knowledge that a lot of readers are putting making good use out of what I print. A case in point: William Linn, of Lithonia, Georgia, was inspired by a combination of several articles' to produce a new software product we all can use!

He calls it AED, which stands for Applesoft EDitor. AED combines in one easy-to-use package:

Line Editing as in PLE and the S-C Macro Assembler
Automatic Line Numbering
Global Search and Replace (with wildcard matching)
Controlled LISTing (Page- or Line-at-a-time, and Slow Scroll)
Display of Variables after execution
Quick entry of DOS commands from a mini-menu
And a lot more.

I said it is easy to use. Why? Here are a few reasons:

The screen is split, with the line being entered at the bottom 6 lines and two possiblities for the top 18 lines. The top 18 lines are used for listing or for display of the most frequently used commands and edit controls.

The commands and edit controls are single letters or control-letters, with mnemonic value.

An inverse letter appears before the prompt character indicating which of six special modes you are in, so you don't get lost.

Clicks and tones provide pleasant feedback at appropriate times.

One very unusual feature, which I have grown to love in a very short time, is a new kind of cursor. Rather than the flashing cursor of the standard Apple input routines, AED alternates the underline character with the character already on the screen. This alternation is done at the same rate as the Apple's flashing mode, but doesn't tire the eyes.

AED loads into memory from $8500 through $95FF, and uses a 256-byte buffer from $8400 to $84FF. HIMEM is set to $83FF.

AED is normally in charge of all input, until the Control-Q command (QUIT) is typed. If you type a letter A, C, E, F, L, M, R, S, or V the rest of the AED command starting with that letter will be displayed. If the command requires no additional information, it is immediately executed. Otherwise, it waits for you to finish the command and type a carriage return. The period is also a command: call it "dot", and think of "DOS", because its purpose is to call up the DOS

Command Mini-Menu. If you type a line beginning with a non-command character, it is passed on to Applesoft. Thus you can enter numbered lines, or type immediate mode commands such as NEW or PRINT X(3) or PR#1.If you do leave AED control, typing "&" will enter AED again. If you have the Autostart Monitor, hitting RESET will re-enter AED.

It is important to realize that you are always in an editing mode. Even commands can be edited using the edit control keys.

Here is a list of the commands:

Letter Commands

```
A AUTO line #,increment
C CHANGE /string1/string2/A
E EDIT line #
F FILE = filename to use in
         DOS commands
L LIST [ line #,line # ]
M MANUAL line numbering
R Repeat last LIST command
S SEARCH /string/
V Variable display
. DOS Mini-Menu
```

Editing Commands

```
^B    Cursor to beginning
^D    Delete a character
^E    Cursor to end of line
^Fx   Cursor to next "x"
^I    Begin Insert mode
^M    (RETURN) Submit line
^N    Cursor to end of line
^R    Recall last line edited
^Tx   Delete through next "x"
^T^T  Delete to end of line
^V    Next character verbatim
^W    Enter word cursor mode
```

Control Commands

```
^A  Assistance
^C  Clear Scroll Area
^Q  Quit
^X  Clear Edit Area
ESC Edit Next Line
```

AED does not have user-defined keyboard macros. The keyboard macros in PLE are a big selling point; however, the ones you actually end up using in PLE are built-in to AED as actual commands or as part of the DOS Mini-Menu. Of course, PLE words with both Integer BASIC and Applesoft; AED is only for Applesoft.

If you use Applesoft, are not already firmly addicted to PLE, and if you do not use Integer BASIC, then you should consider picking up a copy of AED. It is only $40 (same price as PLE), and packs a lot of usefulness for the dollars.

Ashby's Easy Shift-Key Modifier.....Bob Sander-Cederlof

How many times have you read or heard about a way to modify your Apple so that the shift-key would function like a normal typewriter? It is a relatively safe and easy thing to do, but the directions can really be frightening.

Words like "solder", "wire", "take the bottom off your Apple", and so on.

If you have an Apple with the piggy-back board hanging down under your keyboard (Revision 7 or newer), take heart! There is a little device you can pick up for only $15 postpaid, called Ashby's Shift-Key Modifier, which hooks up the modification without any tools or trouble. And it only takes a minute or so! (In fact, only a few seconds if you have done it a few times like I have.)

The Modifier consists of a piece of wire fitted with a plug for the game connector on one end, and with a clip on the other end. The plug is devised so that you still have an empty game socket on top, for attaching paddles or whatever.

To install the Modifier, all you have to do is insert the plug into the game socket, and clip the other end onto the connector from the keyboard to the piggy-back board at the second wire from the right (the RESET key side).

I have installed them on all my Apples, except for my oldest one. (That one is serial #219, bought in August of 1977, and is so old it doesn't even have ventilation slots on the case! Yes, I installed the open-case-and-solder-a-wire modification in the old one.)

Now I can use the shift-key the way I was taught in typing class when I am using Data Capture 4.0, SuperText II, Apple Pie 2.0, the S-C Macro Assembler, or the Word Handler. And more and more programs are being created to take advantage of a REAL shift key on an Apple.

The normal retail price of the Ashby Shift-Key Modifier is $18. I have bought a bunch of them, and you can have them for only $15 each. They come complete with directions for installation.

Potential Trouble in TYMAC.....Robert H. Bernard

[Bob is a subscriber in Westport, Connecticut.]

The article by Peter Bartlett on improving the Epson Controller Card (which appeared in the February 1982 issue of AAL) has prompted me to write to bring to the attention of fellow AAL readers that the TYMAC controller card, which is a lower-cost alternative to the official Epson card, has a potentially serious problem.

To achieve slot independence, controller card ROM programs JSR to an RTS instruction in the Monitor. Then they extract the slot from the return address the JSR put on the stack. The Apple II Reference Manual details the process on page 81-82.

Most controller cards use the Apple technique verbatim, JSR'ing to $FF58, which is an RTS instruction in the Monitor ROM. However, the TYMAC card JSR's to $FDFF. That location also contains an RTS, so there is no problem using the TYMAC card as long as the Monitor ROM is enabled.

The problem occurs when the TYMAC card is used with Pascal. While Apple Computer has specifically guaranteed an RTS instruction at $FF58 in the Pascal Basic Input/Output System (BIOS), no RTS exists at $FDFF. Therefore TYMAC loses control and causes a Pascal crash as soon as it is called.

If any of you have TYMAC cards, and plan to make the Peter Bartlett modification (or perhaps even if you don't plan to), you should also change the JSR instruction at $0A relative to the beginning of the ROM from 20FFFD to 2058FF.

Decision Systems

Decision Systems
P.O. Box 13006
Denton, TX 76203
817/382-6353

DIS-ASSEMBLER

DSA-DS dis-assembles Apple machine language programs into forms compatible with LISA, S-C ASSEMBLER (3.2 or 4.0), Apple's TOOLKIT ASSEMBLER and others. DSA-DS dis-assembles instructions or data. Labels are generated for referenced locations within the machine language program.
$25, Disk, Applesoft (32K, ROM or Language card)

OTHER PRODUCTS

ISAM-DS is an integrated set of Applesoft routines that gives indexed file capabilities to your **BASIC** programs. Retrieve by key, partial key or sequentially. Space from deleted records is automatically reused. Capabilities and performance that match products costing twice as much.
$50 Disk, Applesoft.

PBASIC-DS is a sophisticated preprocessor for structured **BASIC**. Use advanced logic constructs such as **IF...ELSE...**, **CASE**, **SELECT**, and many more. Develop programs for Integer or Applesoft. Enjoy the power of structured logic at a fraction of the cost of **PASCAL**.
$35. Disk, Applesoft (48K, ROM or Language Card).

FORM-DS is a complete system for the definition of input and output froms. **FORM-DS** supplies the automatic checking of numeric input for acceptable range of values, automatic formatting of numeric output, and many more features.
$25 Disk, Applesoft (32K, ROM or Language Card).

UTIL-DS is a set of routines for use with Applesoft to format numeric output, selectively clear variables (Applesoft's **CLEAR** gets everything), improve error handling, and interface machine language with Applesoft programs. Includes a special load routine for placing machine language routines underneath Applesoft programs.
$25 Disk, Applesoft.

SPEED-DS is a routine to modify the statement linkage in an Applesoft program to speed its execution. Improvements of 5-20% are common. As a bonus, **SPEED-DS** includes machine language routines to speed string handling and reduce the need for garbage clean-up. Author: Lee Meador.
$15 Disk, Applesoft (32K, ROM or Language Card).

(Add $4.00 for Foreign Mail)

***Apple II is a registered trademark of the Apple Computer Co.**

Using Macros and Nested Macros.....Art Schumer

[Art is a subscriber in Manvel, North Dakota; he is the programming side of S&H Software. Art wrote the Universal Boot Initializer, The DOS Enhancer, and the AmperCat Utility.]

The new S-C Macro Assembler is truly the best assembler around. With the addition of Macros, easier programming is limited only by your imagination. All you have to do is dream up some uses for Macros. Are Macros and Nested Macros really worth using? You bet! One of my source files was 104 sectors long, but after going back through it and implementing macros, the file shortened to only 96 sectors; it was also easier to read.

As Bob pointed out in the manual, nested macros are allowed in this new version, but he frowned on their use. I beg to differ with him, as I believe that nested macros can make your source files easier to read, as well as easier to write. They may seem complex at first, but after setting them up they become very easy to use.

In my example program, I've defined a macro called GOTO.XY that will take two variables and use them to position the cursor. Another defined macro called CLEAR.XY is a singly nested macro that uses GOTO.XY to position the cursor, and then clears from there to the end of screen. CLEAR.PRINT.XY positions the cursor (using GOTO.XY inside CLEAR.PRINT.XY), clears the rest of the screen, and prints a message. It may sound confusing, but after examining the source listing and th macro definitions, it should be easy to understand how this all works.

In all the macros, the first variable is the horizontal cursor position and the second variable is the vertical cursor position. CLEAR.PRINT.XY calls on a subroutine (JSR PRNT), which expects the message to follow the JSR instruction. The message is terminated by a 00 byte, and execution proceeds at the instruction which follows the message in memory.

The PRNT subroutine came from a Call A.P.P.L.E. article by Andy Hertzfeld.

Have fun with your new S-C Macro Assembler!

```
                1000 *-------------------------------
                1010 * USE OF MACROS & NESTED MACROS
                1020 * BY ART SCHUMER - 3/25/82
                1030 *-------------------------------
FB5B-           1040 VTAB    .EQ $FB5B
FC42-           1050 CLREOP  .EQ $FC42
FC58-           1060 HOME    .EQ $FC58
FD0C-           1070 RDKEY   .EQ $FD0C
FDED-           1080 COUT    .EQ $FDED
                1090 *-------------------------------
0006-           1100 PTR     .EQ $6
0024-           1110 CH      .EQ $24
0025-           1120 CV      .EQ $25
```

QUICKTRACE

relocatable program traces and displays the actual machine operations, *while* it is running without interfering with those operations. Look at these *FEATURES:*

***Single-Step** mode displays the last instruction, next instruction, registers, flags, stack contents, and six user-definable memory locations.*

***Trace** mode gives a running display of the Single-Step information and can be made to stop upon encountering any of nine user-definable conditions.*

***Background** mode permits tracing with no display until it is desired. Debugged routines run at near normal speed until one of the stopping conditions is met, which causes the program to return to Single-Step.*

***QUICKTRACE** allows changes to the stack, registers, stopping conditions, addresses to be displayed, and output destinations for all this information. All this can be done in Single-Step mode while running.*

***Two optional display formats** can show a sequence of operations at once. Usually, the information is given in four lines at the bottom of the screen.*

***QUICKTRACE** is completely transparent to the program being traced. It will not interfere with the stack, program, or I/O.*

***QUICKTRACE** is relocatable to any free part of memory. Its output can be sent to any slot or to the screen.*

***QUICKTRACE** is completely compatible with programs using Applesoft and Integer BASICs, graphics, and DOS. (Time dependent DOS operations can be bypassed.) It will display the graphics on the screen while **QUICKTRACE** is alive.*

***QUICKTRACE** is a beautiful way to show the incredibly complex sequence of operations that a computer goes through in executing a program*

QUICKTRACE *$50*

Is a trademark of Anthro-Digital, Inc.

Written by John Rogers

See these programs at participating Computerland and other fine computer stores.

Anthro - Digital Software, Inc.
P.O. Box 1385 Pittsfield, MA 01202

```
                        1130 *--------------------------------
                        1140 *       MACRO DEFINITIONS
                        1150 *
                        1160 * CLR.PRNT.XY AND GOTO.PRNT.XY
                        1170 * ARE EXAMPLES OF NESTED MACROS
                        1180 *--------------------------------
                        1190         .MA GOTO.XY
                        1200         LDA #]1
                        1210         STA CH
                        1220         LDA #]2
                        1230         JSR VTAB
                        1240         .EM
                        1250 *--------------------------------
                        1260         .MA CLEAR.XY
                        1270         >GOTO.XY ]1,]2
                        1280         JSR CLREOP
                        1290         .EM
                        1300 *--------------------------------
                        1310         .MA CLEAR.PRNT.XY
                        1320         >CLEAR.XY ]1,]2
                        1330         JSR PRNT
                        1340         .EM
                        1350 *--------------------------------
                        1360         .MA GOTO.PRNT.XY
                        1370         >GOTO.XY ]1,]2
                        1380         JSR PRNT
                        1390         .EM
                        1400 *--------------------------------
                        1410         .MA READ.XY
                        1420         >GOTO.XY ]1,]2
                        1430         JSR RDKEY
                        1440         .EM
                        1450 *--------------------------------
                        1460 *       THE PROGRAM . . . .
                        1470 *--------------------------------
0800- 20 58 FC          1480 START   JSR HOME
0803-                   1490         >GOTO.PRNT.XY 4,12
0803-                   0000>         >GOTO.XY 4,12
0803- A9 04             0000>>         LDA #4
0805- 85 24             0000>>         STA CH
0807- A9 0C             0000>>         LDA #12
0809- 20 5B FB          0000>>         JSR VTAB
080C- 20 6A 08          0000>         JSR PRNT
080F- D4 C8 C9
0812- D3 A0 C5
0815- D8 C1 CD
0818- D0 CC C5
081B- A0 D5 D3
081E- C5 D3 A0
0821- CE C5 D3
0824- D4 C5 C4
0827- A0 CD C1
082A- C3 D2 CF
082D- D3                1500         .AS -/THIS EXAMPLE USES NESTED MACROS/
082E- 00                1510         .HS 00
082F-                   1520         >READ.XY 36,12
082F-                   0000>         >GOTO.XY 36,12
082F- A9 24             0000>>         LDA #36
0831- 85 24             0000>>         STA CH
0833- A9 0C             0000>>         LDA #12
0835- 20 5B FB          0000>>         JSR VTAB
0838- 20 0C FD          0000>         JSR RDKEY
083B-                   1530         >CLEAR.PRNT.XY 4,12
083B-                   0000>         >CLEAR.XY 4,12
083B-                   0000>>         >GOTO.XY 4,12
083B- A9 04             0000>>>         LDA #4
083D- 85 24             0000>>>         STA CH
083F- A9 0C             0000>>>         LDA #12
0841- 20 5B FB          0000>>>         JSR VTAB
0844- 20 42 FC          0000>>         JSR CLREOP
0847- 20 6A 08          0000>         JSR PRNT
084A- C1 CE C4
084D- A0 D4 C8
0850- C9 D3 A0
0853- CF CE C5
0856- A0 C1 CC
0859- D3 CF A1          1540         .AS -/AND THIS ONE ALSO!/
085C- 00                1550         .HS 00
```

SUPER PHONE

For Applesoft and D. C. HAYES MICROMODEM II users only

Machine language terminal program

Capture data in big buffer or not

READ or BSAVE or PRINT the buffer

Keep a telephone directory

Dial automatically from directory

Call persons or modems

Capture programs from an unattended remote Apple

Send programs to an unattended remote Apple

Time a call

Transmit or receive machine code, shape tables, programs and data

Answers phone or picks up

Terminal works with THE SOURCE and all bulletin boards I've tried.

SUPER PHONE is delivered by phone only. It is all machine language but you must have Applesoft up to get it or use it. Delivering it this way proves it works. I am not selling it on disk or tape.

Almost everything is done from a menu. If it won't do as I say above I'll be glad to give you a refund but I'll deduct the cost of the phone call. It takes about 20 minutes to put it in your Apple. I make calls any time.

I have to operate as the phone company does. For night and weekend delivery send $25.00. For evening delivery send $27.00. For weekday delivery send $30.00.

I send the short documentation you need in the mail. Let me know time of day to call.(Eastern Time.) First we'll talk, then I'll put it in your memory and BSAVE it. Then we'll talk. You don't have to do anything except follow a couple of instructions. You can watch it enter your memory.

The reason I am selling for this low price right now is that I want input as to what can be done to improve it before I put it on the market, generally. A program which will do what I claim for it is worth a lot more than $25.00. Obviously, I can't distribute many copies this way. I will have to shut it off when it gets onerous. First come. First served. I refund money if I don't deliver in two weeks. Don't call me. I'll call you.

Send check, address, phone number to:

James O. Church
20 Placid Street
Trumbull, Ct. 06611

```
085D-            1560         >READ.XY 22,12
085D-            0000>         >GOTO.XY 22,12
085D- A9 16      0000>>         LDA #22
085F- 85 24      0000>>         STA CH
0861- A9 0C      0000>>         LDA #12
0863- 20 5B FB   0000>>         JSR VTAB
0866- 20 0C FD   0000>         JSR RDKEY
0869- 60         1570         RTS
                 1580 *-------------------------------
                 1590 * ANDY HERTZFELD'S PRINT ROUTINE
                 1600 *-------------------------------
086A- 68         1610 PRNT    PLA
086B- 85 06      1620         STA PTR
086D- 68         1630         PLA
086E- 85 07      1640         STA PTR+1
0870- A0 00      1650         LDY #0
0872- E6 06      1660 .1      INC PTR
0874- D0 02      1670         BNE .2
0876- E6 07      1680         INC PTR+1
0878- B1 06      1690 .2      LDA (PTR),Y
087A- F0 06      1700         BEQ .3
087C- 20 ED FD   1710         JSR COUT
087F- 4C 72 08   1720         JMP .1
0882- A5 07      1730 .3      LDA PTR+1
0884- 48         1740         PHA
0885- A5 06      1750         LDA PTR
0887- 48         1760         PHA
0888- 60         1770         RTS
                 1780 *-------------------------------
```

Recursive Macro Example.....Lee Meador

[Lee is a subscriber from Arlington, Texas. He wrote the original code for the .TF directive and REPLACE command in the S-C Assemblers.]

Here is short example of a useful macro that uses a recursive definition. By recursive I mean that the definition calls itself.

Most large computers have a shift instruction which can shift any number of bits; the 6502 shifts only shift one bit at a time. The LSR macro shown here accepts a shift count as the first parameter, and generates one LSR opcode for each bit shift you want.

The second parameter is optional. If there is no second parameter, the A-register will be shifted. If you specify a variable for the second parameter, that memory location will be shifted. Both cases are shown in the example below.

How does it work? The definiton says to test the first parameter; if it is greater than zero, generate the LSR with the optional second parameter as the address field, and call on the LSR macro with the first parameter decremented by one. If the first parameter is zero (and it eventually will be), no code is generated. Read the listing carefully, noting the indentation, and you should be able to follow it.

```
                        1000        .MA LSR
                        1010        .DO ]1>0
                        1020        LSR ]2
                        1030        >LSR ]1-1,]2
                        1040        .FIN
                        1050        .EM
                        1060 *--------------------------------
0800-                   1070        >LSR 3,$12
                        0000>        .DO 3>0
0800- 46 12             0000>        LSR $12
0802-                   0000>        >LSR 3-1,$12
                        0000>>        .DO 3-1>0
0802- 46 12             0000>>        LSR $12
0804-                   0000>>        >LSR 3-1-1,$12
                        0000>>>        .DO 3-1-1>0
0804- 46 12             0000>>>        LSR $12
0806-                   0000>>>        >LSR 3-1-1-1,$12
                        0000>>>>        .DO 3-1-1-1>0
                        0000>>>>        .FIN
                        0000>>>        .FIN
                        0000>>        .FIN
                        0000>        .FIN
0806-                   1100        >LSR 2
                        0000>        .DO 2>0
0806- 4A                0000>        LSR
0807-                   0000>        >LSR 2-1,
                        0000>>        .DO 2-1>0
0807- 4A                0000>>        LSR
0808-                   0000>>        >LSR 2-1-1,
                        0000>>>        .DO 2-1-1>0
                        0000>>>        .FIN
                        0000>>        .FIN
                        0000>        .FIN
```

INTRODUCING AED][

A POWERFUL AID FOR APPLESOFT PROGRAM DEVELOPMENT

- A full function line editor supports character insertion, single and multiple character deletion, line truncation, instant cursor positioning to a specific character, the beginning, or end of any APPLESOFT line.
- All editing functions are in effect while entering new APPLESOFT statements as well as revising existing lines of your Basic programs.
- Search your APPLESOFT program for a particular string and immediately begin editing the line with the cursor positioned on the matching string.
- Global search and replace one string with another in all or selective lines.
- Wildcard characters supported in searching: /#TAB/ matches HTAB, VTAB, etc.
- Automatic line numbering with user specified increment.
- Sophisticated LIST control allows page-at-a-time, line-at-a-time, fast and slow speed scrolling of APPLESOFT lines. Say goodbye to CTRL-S and CTRL-C. Single key repeat of last LIST command eliminates retyping line number range.
- AED remains resident while RUNning your APPLESOFT program. Single key-stroke exit from AED; reentry to AED via & (ampersand) or pressing RESET.
- A single key command displays the names and values of each APPLESOFT simple variable (very useful in debugging).
- DOS command menu provides CATALOG (D1 or D2), LOAD, SAVE, LOCK, UNLOCK, or DELETE of the current program via one or two keystrokes.
- Direct keyboard entry of [,], \, and _ characters.
- Single key recall of HELP screen, even while editing a line.
- Careful attention to human factors: blinking underscore cursor, soft bell tone for errors, visual mode indicator, and minimal keystroke commands.
- AED is 100% Assembly Language and uses approximately 4K RAM.
- System Requirements: Apple][with 48K RAM, Applesoft ROM, and DOS 3.3

AED][DISKETTE & USER GUIDE:.....$40.00

DEALER INQUIRIES INVITED

LINN SOFTWARE
3199 HAMMOCK CREEK
LITHONIA, GEORGIA 30058

Controlling Software Configuration.....Don Taylor

Paul Schlyter's article on moving the S-C Assembler into the language card (AAL January 1982) couldn't have come at a better time for me. I was working on a project that had just outgrown the available memory space, and LANGASM came to the rescue. Long live LANGASM!

LANGASM and the extensions to the S-C Assembler that have appeared in the AAL bring to the fore an important subject: controlling the configuration of your copy of someone else's software.

How do I know that a particular "patched" copy I have of the assembler is compatible with another extension that will appear in next month's AAL? What kind of documentation must I keep somewhere to keep track of patched object code for which I have no source code? And how many different patched source code versions (to which I have given different names) of the S-C Assembler am I willing to keep track of?

For my use, I've chosen to keep track of only two modified copies of the assembler; I call them ASM II.1 and LANGASM.1. These two versions are simply the "standard issue" S-C Assembler Version 4.0 and LANGASM, each augmented with the listed .DA directive patch described by Bob in the December, 1980 issue of AAL. (I chose this configuration because the extension was written by Bob himself, and because other AAL articles have used the listed .DA directive. The feature is upward compatible, and listed .DAs presented to unmodified copies of the assembler will cause invisible errors by seemingly accepting those directives, while generating no code for items betond the comma.)

To add the extensions I want, I first load in ASM II.1 or LANGASM.1, and then modify the copy in memory with a configuration file before using it.

The source listing of LANGASM.1 EXT.SRC shows the method I use to add HOME, COPY and EDIT commands to my copy of LANGASM.1. This particular routine is .OR'd at the beginning of one of the 4K language card memory blocks located at $D000, which permits several extensions to be loaded in one contiguous area of memory, while leaving the main memory area free for the source file and symbol table.

Lines 1160-1570 install the patches in the memory-resident copy of LANGASM.1 and then return to a calling routine. Lines 1320-1400 patch the FAST command (disabled by the LANGASM patches) to render it a HOME command that works like Applesoft's does.

Lines 1260-1430 make similar modifications to LANGASM's command table entries, replacing LOAD with COPY and SAVE with EDIT, along with their assembled addresses (less one).

Lines 1440-1520 are the patches that were contained in Mike Laumer's source code for the EDIT command, found in the January, 1981 issue of AAL.

The source files for EDIT and COPY used within LANGASM.1 EXT.SRC in lines 1590 to the end of the file are identical to those written by Mike Laumer and Bob Sander-Cederlof, with a couple of exceptions. As stated above, the patch code for NML was moved to the modification area in lines 1440-1520. Second, all .OR and .TF directives were removed from both files. Third, a few redundant .EQ directives (internal assembler reference addresses) had to be removed to avoid any EXTRA DEFINITION errors. Finally, $D000 was added to all internal assembler references to make them compatible with LANGASM's $E000 origin.

To install these patches to LANGASM.1, I EXEC the following text file, which I call LANGASM:

```
CALL -151              (get into the monitor)
C0C1                   (turn off any firmware card)
C081 C081              (write enable the language card)
BLOAD LANGASM.1        (load LANGASM into the language card)
BLOAD MONITOR EXTENSIONS (load in page 3 extensions
                          from 10/81 issue of AAL)
BLOAD LANGASM.1 EXTENSIONS (load in the mods)
A5B8:80                (patch DOS to use the language card)
A5C0:81
300G                   (install monitor extensions)
C083                   (switch in Bank 2)
D000G                  (install LANGASM mods)
3D3G                   (return to DOS and Applesoft)
INT                    (enter the assembler)
```

To use this method of in-memory configuration with ASM II.1 (where patches can't always be added in contiguous memory), I use a separate file for each command patch, each .OR'd at the proper address, and then install all patch routines within a single text file that is EXEC'd. Since I'm not dealing with the language card, and each of the commands added above are indepedent of one another, I can skip the EXEC and just BLOAD and install each command (or group of commands) I want to add with the monitor. The result is an easy configuration of the assembler, done at run time.

The use of configuration files to modify the assembler takes a few extra seconds (and a couple of extra files on my utility disk), but it is no more work thanks to the EXEC file. It permits me to keep only a single copy of the assembler (in a known configuration), while enabling me to fully document any modifications I make to the assembler with configuration files for which I have the source code. By creating different EXEC files, I can quickly and easily intermix configuration files to create (and document!) any version of the assembler I wish.

Even though I suppressed the listing of the EDIT and COPY commands to save newsletter space, the source code is on the Quarterly Disk (#7) which will include this program.

```
                    1000 *--------------------------------
                    1010 *       INSTALL EXTENSIONS TO LANGASM
                    1020 *
                    1030 *       AUTHOR:  DON TAYLOR
                    1040 *         DATE:  2/6/82, 4:00 PM
                    1050 *
                    1060 *--------------------------------
                    1070         .OR $D000
                    1080         .TF LANGASM.1 EXTENSIONS
                    1090 *--------------------------------
03D0-               1100 DOS.REENTRY              .EQ $03D0
FC58-               1110 MON.HOME                 .EQ $FC58
E246-               1120 SCA.LOAD.CMD             .EQ $E246
E26E-               1130 SCA.SAVE.CMD             .EQ $E26E
E273-               1140 SCA.SLOW.CMD             .EQ $E273
                    1150 *--------------------------------
                    1160 INSTALL.MODIFICATIONS
D000- A0 02         1170         LDY #2             MODIFY ASSEMBLER
D002- B9 57 D0      1180 .1      LDA HOME.TABLE,Y   COMMAND JUMP
D005- 99 73 E2      1190         STA SCA.SLOW.CMD,Y
D008- 88            1200         DEY
D009- 10 F7         1210         BPL .1
D00B- A9 57         1220         LDA #MON.HOME-1
D00D- 8D 76 E2      1230         STA SCA.SLOW.CMD+3
D010- A9 FC         1240         LDA /MON.HOME-1
D012- 8D 77 E2      1250         STA SCA.SLOW.CMD+4
D015- A0 02         1260         LDY #2
D017- B9 5A D0      1270 .2      LDA COPY.TABLE,Y
D01A- 99 46 E2      1280         STA SCA.LOAD.CMD,Y
D01D- 88            1290         DEY
D01E- 10 F7         1300         BPL .2
D020- A9 65         1310         LDA #COPY-1
D022- 8D 49 E2      1320         STA SCA.LOAD.CMD+3
D025- A9 D0         1330         LDA /COPY-1
D027- 8D 4A E2      1340         STA SCA.LOAD.CMD+4
D02A- A0 02         1350         LDY #2
D02C- B9 5D D0      1360 .3      LDA EDIT.TABLE,Y
D02F- 99 6E E2      1370         STA SCA.SAVE.CMD,Y
D032- 88            1380         DEY
D033- 10 F7         1390         BPL .3
D035- A9 43         1400         LDA #EDIT-1
D037- 8D 71 E2      1410         STA SCA.SAVE.CMD+3
D03A- A9 D1         1420         LDA /EDIT-1
D03C- 8D 72 E2      1430         STA SCA.SAVE.CMD+4
D03F- A9 60         1440         LDA #$60         PATCH NML TO
D041- 8D 25 E1      1450         STA $E125        MAKE IT A
D044- A9 4C         1460         LDA #$4C         SUBROUTINE
D046- 8D 63 E0      1470         STA NML
D049- 8D 78 E0      1480         STA $E078
D04C- A9 2C         1490         LDA #NEW.NML
D04E- 8D 64 E0      1500         STA NML+1
D051- A9 D1         1510         LDA /NEW.NML
D053- 8D 65 E0      1520         STA NML+2
D056- 60            1530         RTS
                    1540 *--------------------------------
D057- 48 4F 4D      1550 HOME.TABLE .AS ^HOM^
D05A- 43 4F 50      1560 COPY.TABLE .AS ^COP^
D05D- 45 44 49      1570 EDIT.TABLE .AS ^EDI^
                    1580 *--------------------------------
                    1590 * COPY COMMAND FOR S-C ASSEMBLER
                    1600 * VERSION 4.0
                    1610 *
                    1620 * SOURCE: BOB SANDER-CEDERLOF 12/80
                    1630 *
                    1640 *--------------------------------
                    1650 *
                    1660 *
                    1670 * NOTE: COPY FUNCTION SOURCE IS
                    1680 *       ASSEMBLED HERE...

                    3130 *
                    3140 *
                    3150 * NOTE: EDIT FUNCTION SOURCE IS
                    3160 *       ASSEMBLED HERE...
```

```
                        1000 *--------------------------------
                        1010 *       FUNNY NOISE
                        1020 *--------------------------------
C030-                   1030 SPKR    .EQ $C030     SPEAKER TOGGLE ADDRESS
C000-                   1040 KYBD    .EQ $C000     KEYBOARD INPUT
C010-                   1050 STROBE  .EQ $C010     KEYBOARD STROBE
                        1060 *--------------------------------
0000-                   1070 PNTR    .EQ 0         ADDRESS OF CURRENT RANDOM VALUE
                        1080 *--------------------------------
0800- 20 58 FC          1090 NOISE   JSR $FC58     CLEAR SCREEN, HOME CURSOR
0803- A0 00             1100 N0      LDY #0        POINT TO FIRST BYTE IN PAGE
0805- A9 00             1110         LDA #$D000    START AT $D000
0807- 85 00             1120         STA PNTR
0809- A9 D0             1130         LDA /$D000
080B- 85 01             1140         STA PNTR+1
080D- 20 DA FD          1150         JSR $FDDA     PRINT PAGE NUMBER
0810- AD 30 C0          1160 N1      LDA SPKR      TOGGLE SPEAKER
0813- B1 00             1170         LDA (PNTR),Y  GET HALF-CYCLE TIMER
0815- AA                1180         TAX
0816- CA                1190 N2      DEX           DELAY LOOP FOR HALF-CYCLE
0817- D0 FD             1200         BNE N2
0819- C8                1210         INY           NEXT BYTE IN PAGE
081A- D0 F4             1220         BNE N1
081C- E6 01             1230         INC PNTR+1    NEXT PAGE
081E- A5 01             1240         LDA PNTR+1    BYPASS I/O AREA
0820- C9 C0             1250         CMP /$C000
0822- F0 DF             1260         BEQ N0
0824- 20 DA FD          1270         JSR $FDDA     PRINT PAGE NUMBER
0827- AD 00 C0          1280         LDA KYBD      SEE IF ANY KEY PRESSED
082A- 10 E4             1290         BPL N1        NO, KEEP MAKING NOISE
082C- 8D 10 C0          1300         STA STROBE    YES, CLEAR STROBE
082F- 60                1310         RTS           THAT'S ALL, FOLKS!
```

Apple Assembly Line is published monthly by S-C SOFTWARE CORPORATION, P. O. Box 280300, Dallas, Texas 75228. Phone (214) 324-2050. Subscription rate is $15 per year in the USA, sent Bulk Mail; $18 per year sent First Class Mail in USA, Canada, and Mexico; $28 per year sent Air Mail to other countries. Back issues are available for $1.50 each (other countries add $1 per back issue for postage).